```
class Node:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

def create_sample_tree():
    root = Node(1)
    root.left = Node(2)
    root.right = Node(3)
    root.left.left = Node(4)
    root.left.right = Node(5)
    root.right.left = Node(6)
    root.right.right = Node(7)
    return root

def is_terminal_node(node):
    return not (node.left or node.right)

def evaluate_node(node):
    return node.value

def get_children(node):
    return [child for child in [node.left, node.right] if child]

def minimax(node, depth, maximizingPlayer):
    if depth == 0 or is_terminal_node(node):
        return evaluate_node(node)

    if maximizingPlayer:
        maxEval = float('-inf')
```

```
        for child in get_children(node):
            eval_child = minimax(child, depth - 1, False)
            maxEval = max(maxEval, eval_child)
        return maxEval
    else:
        minEval = float('inf')
        for child in get_children(node):
            eval_child = minimax(child, depth - 1, True)
            minEval = min(minEval, eval_child)
        return minEval

# Example usage:
sample_tree = create_sample_tree()
result = minimax(sample_tree, depth=3, maximizingPlayer=True)
print("Result:", result)
```